**HashMapper: Text Fingerprinting System**

**Comprehensive Documentation**

# Introduction

HashMapper is a unique text-to-image fingerprinting system that transforms input text into colorful visual representations based on hash collisions.   
Rather than minimizing collisions as in typical hash functions, this system intentionally creates and visualizes collisions to explore how poor hashing behaviors affect data storage and distribution.   
This project combines Python (Flask) for the frontend and server control, and Java for the backend hash map logic and visualization.

# Key Definitions

## Text Fingerprinting

The process of generating a compact, recognizable representation of a text input. In HashMapper, fingerprinting means creating a visual map that captures how the text's characters and words interact under hashing.

## Hash Function

A mathematical function that takes an input and produces a fixed-size string or number, which is used for fast data retrieval. Good hash functions spread data evenly across buckets, but HashMapper intentionally explores 'bad' hashing.

## Hash Map

A data structure used to store key-value pairs efficiently. Hash functions determine where keys and values are stored.

## Collision

When two different inputs generate the same hash code, a collision occurs. In HashMapper, these collisions are deliberately exaggerated to create more interesting visualization patterns.

## Bucket

A container in a hash map that holds one or more key-value pairs. If multiple inputs hash to the same bucket, they are stored together.

## DumbHashMap

A simplified hash map with intentionally poor hashing behavior to promote collisions. Used in HashMapper for educational and visualization purposes.

## Text Visualization

Graphically representing patterns or structures in text data. In HashMapper, text visualization shows how characters and words distribute and collide when hashed.

# Features

- Visualization of hash collisions for text data  
- Simple DumbHashMap and SimpleHashMap implementations  
- Python-to-Java bridge (`java\_bridge.py`) to integrate Java with Python Flask server  
- CSV generation for hash analysis (collisions, distributions, lookup performance)  
- Interactive web interface built with Flask  
- Ready-to-use shell scripts for Java compilation and project setup

# Installation

Prerequisites:

- Python 3.8+  
- Java 8 or higher  
- pip packages: Flask, Pillow

1. Setup Steps:

* 1. Clone or unzip the project files.
* 2. (Optional) Create a virtual environment.
* 3. Install dependencies using pip: `pip install flask Pillow`.
* 4. Compile Java classes: `./compile\_java.sh` (Linux/Mac) or `compile\_java.bat` (Windows).

# Usage Instructions

## Command Line

To run the Flask server:  
$ python app.py  
  
Then, open a browser and navigate to: http://localhost:5000  
  
You can enter text into the web app to generate its fingerprint visualization and view collision analysis tables.

## Python API

You can also directly call the Java backend from Python:  
  
Example:  
from java\_bridge import generate\_fingerprint  
  
raw\_img, enhanced\_img, stats = generate\_fingerprint(  
 text="Hello world!",  
 size=512,  
 hash\_function="SimpleHashMap",  
 salt\_level=2,  
 smooth\_radius=3  
)

## Java CLI

You can manually test the Java fingerprint generator via command line:  
  
$ java -cp "lib/\*:.:java" HashMapper --text-file input.txt --size 512 --hash-function SimpleHashMap

# Architecture

System Components:  
- Frontend: Flask + HTML templates  
- Backend Controller: `java\_bridge.py`  
- Backend Core Logic: Java classes (`HashMapper.java`, `SimpleHashMap.java`, etc.)  
- Visualization Library: JFreeChart (through lib/)

Data Flow:  
User Input → Flask App → Python Controller → Java Program → Hash Analysis → Image/CSV Output → Flask Response

# File Structure

HashMapper-txt2img-main/  
├── app.py # Flask server  
├── java\_bridge.py # Python to Java bridge  
├── java/ # Java source files (HashMapper.java, SimpleHashMap.java, etc.)  
├── lib/ # Java external libraries (JFreeChart)  
├── static/ # Static files (CSS, images)  
├── templates/ # HTML templates  
├── \*.csv # Generated analysis files  
├── compile\_java.sh / .bat # Compilation scripts  
├── run.sh # Quick run script  
└── README.md # Project overview

# Outputs and Results

- bucket\_distribution.csv: Distribution of entries across hash buckets  
- string\_collisions.csv: How many string collisions occurred  
- lookup\_performance.csv: Average lookup times  
- text\_fingerprint\_analysis.csv: General analysis of text fingerprinting  
- Generated Images: Fingerprint visualization PNGs

# License

This project is made available for educational and experimental purposes. Redistribution and modification are permitted under open-source licenses. For commercial use, please consult the original author(s).